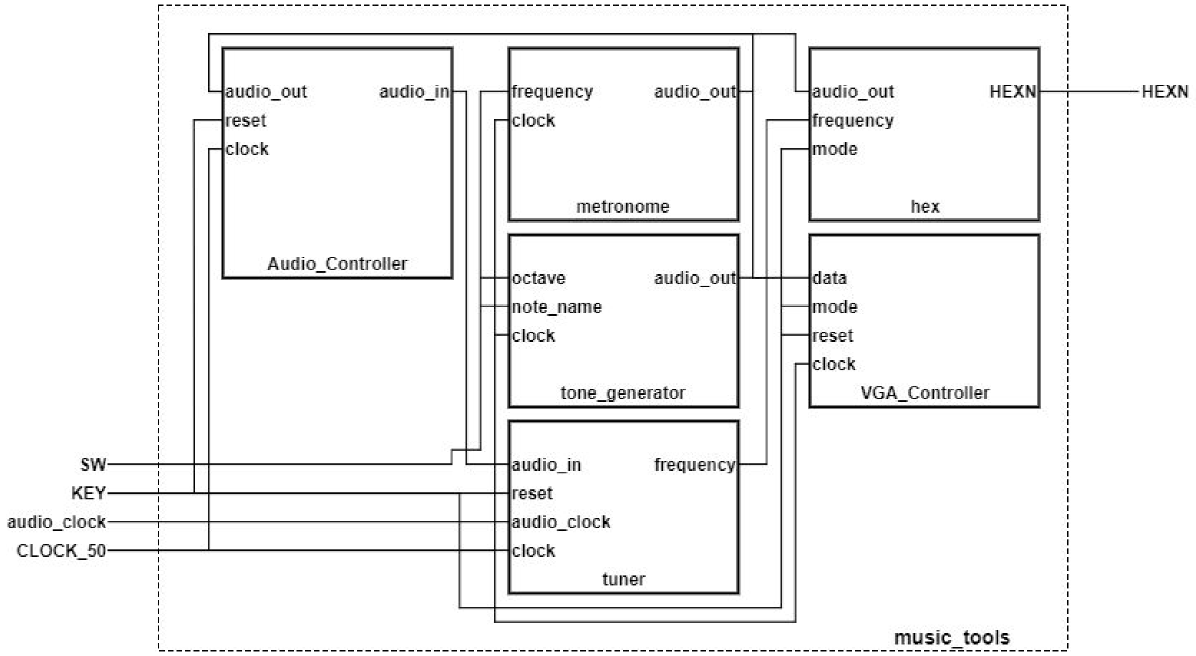
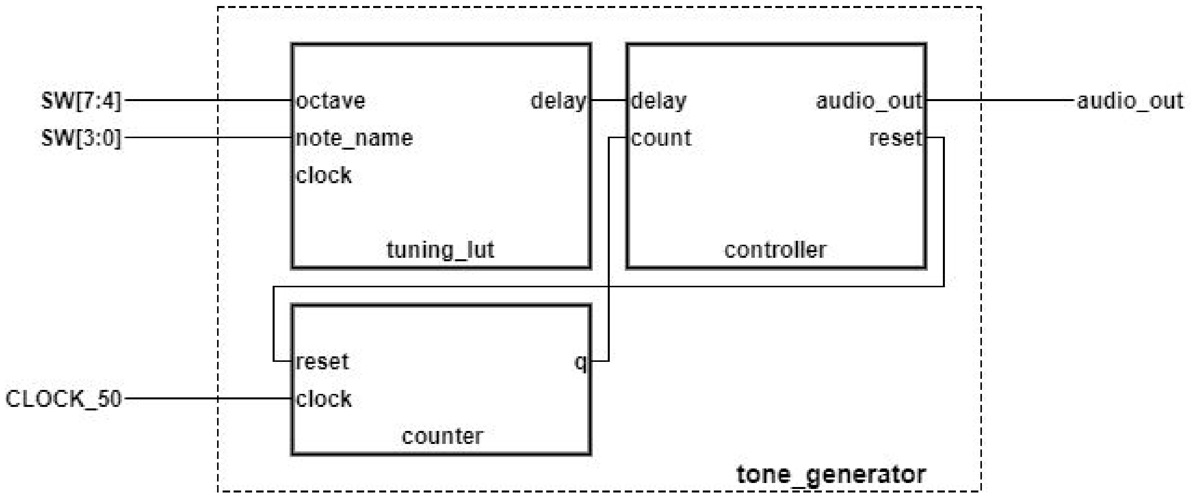
Introduction

This project is targeted towards musicians and aims to provide a suite of simple tools, consisting of a tone generator, a metronome, and a tuner. Both of us have experience with musical instruments, so we found it as common ground and motivation for this project.

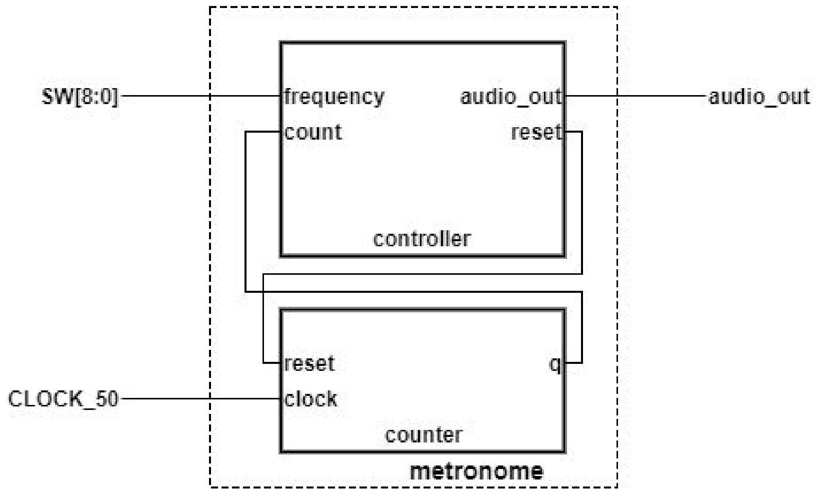
Design



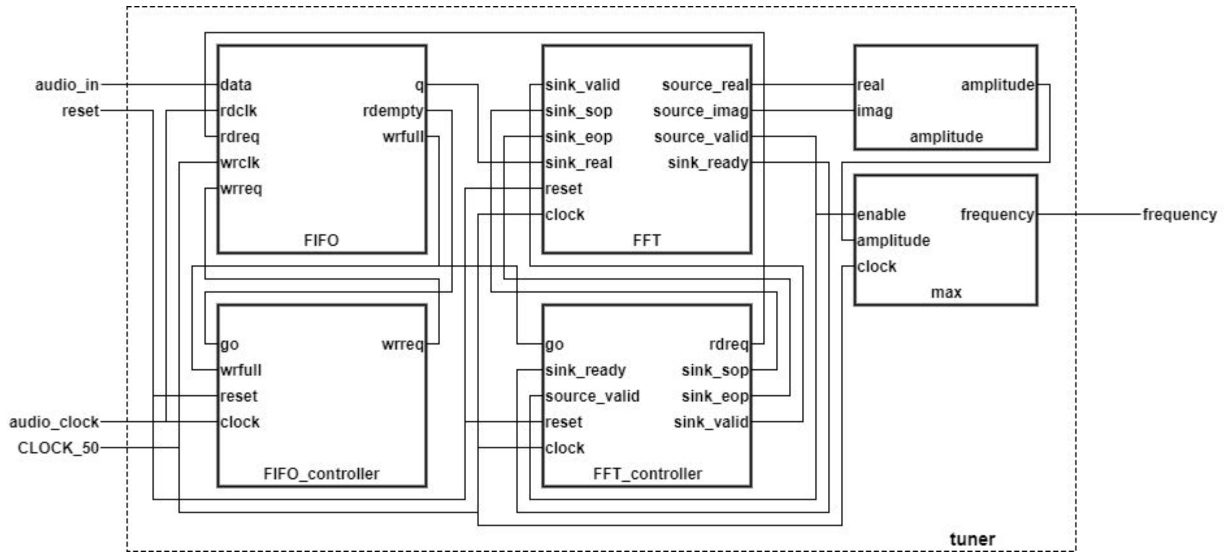
At the top level, the design uses mode buttons to switch between the tone generator, metronome, and tuner functionalities. The interface between these functionalities, the audio controller, the hex displays, and the VGA output is also done in the top level. A small FSM (shown in full schematic in Appendix A) is used to send the redraw signal to vga to update sprite locations, as well as reset the audio core (otherwise abrupt changes in sound output could cause the audio core to freeze).



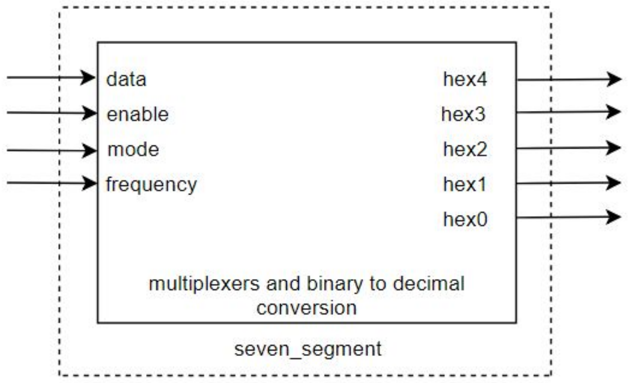
The tone generator functionality is selected when the mode select is 1. This tool receives a binary input from the switches (octave number from SW3-0, note name from SW7-4) on the FPGA and interprets it as a note (e.g. A4). This information is then converted to a frequency via a lookup table, which is subsequently used to generate a square wave audio signal that is transmitted by the audio controller. The note that is selected and played is displayed graphically using VGA output.



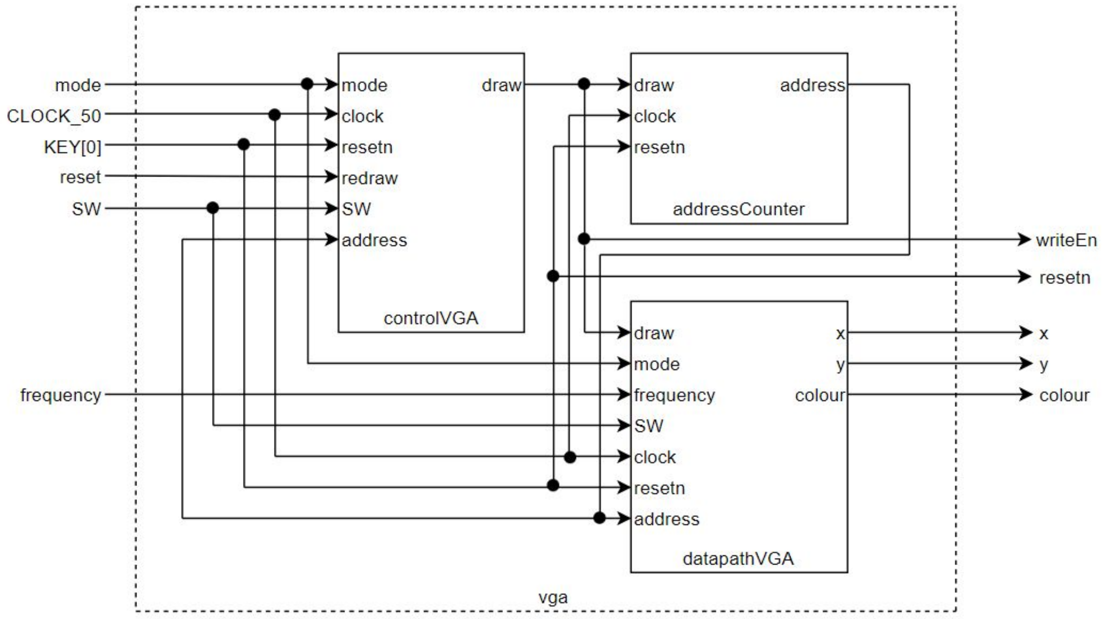
The metronome functionality is selected when the mode select is 2. This tool, like the tone generator, receives a binary input from the FPGA switches 8-0. This input is interpreted as the frequency of the metronome ticks in beats per minute, which is then used to generate an appropriate audio signal. Again, the VGA output is used to display graphically the selected frequency of the metronome.



The tuner functionality is selected when the mode select is 3. Audio input is received from the audio controller, which is then passed through a Fast Fourier Transform. The outputs of the transform are read, and the frequency with the highest amplitude (i.e. the dominant frequency) is shown on the hex display. In addition, the dominant frequency is passed through a lookup table, which determines the musical note closest to that frequency as well as the frequency offset from that note. This information is displayed graphically using VGA output.



This module is used to display information on the HEX displays. The full schematic is in Appendix B. All numeric values are converted to a base 10 number. In tone generator mode, it displays the octave number (selected by SW3-0) on HEX4 and the note names (selected by SW7-4) on HEX3-0. A whole tone is displayed like “A” and a semitone is displayed like “AHBb”. If an invalid octave and note combination is selected, “Error” is displayed. In metronome mode, the bpm value is displayed on HEX2-0. In tuner mode, the frequency in Hz is displayed on HEX4-0.



The vga module sends signals (pixel coordinates, pixel colour) to the VGA core to draw the desired image on the monitor screen one pixel at a time (resolution is 320x240, 3 bit colour). The control uses an FSM to instruct the datapath to draw the background or sprites, as well as reset the counter. The datapath contains RAM modules for all backgrounds and sprites. It uses the draw signal and counter value from control, as well as external inputs (mode, switches and frequency) to load data from RAM and send out coordinates and colours. Detailed schematics are found in Appendix C and D.

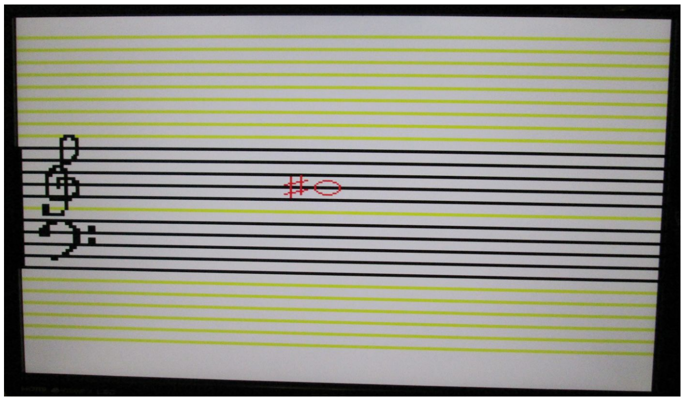
In tone generator mode, the background is a music staff, with treble clef, bass clef and additional lines to show notes from A0 to C8. When the note and octave inputs are valid, a whole note sprite is loaded and drawn to indicate the tone being generated. For semitones, it also draws a sharp sign sprite left of the note.

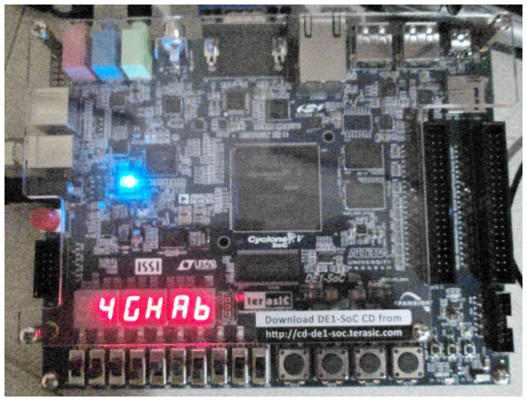
In metronome mode, the background consists of a scale spanning the width of the screen, each pixel representing 1 bpm. For example, if the bpm input is 100, a red line is drawn on the 100th pixel from the left of the screen on the scale.

In tuner mode, the background shows a scale indicating the frequency offset from the closest music note. 0 Hz is in the middle. Each pixel left or right of the middle indicates 1 Hz deviation negative or positive respectively. A lookup table is used to select the octave number sprite (drawn above the scale) and note name sprite (drawn below the scale) of the closest music note to the given frequency value.

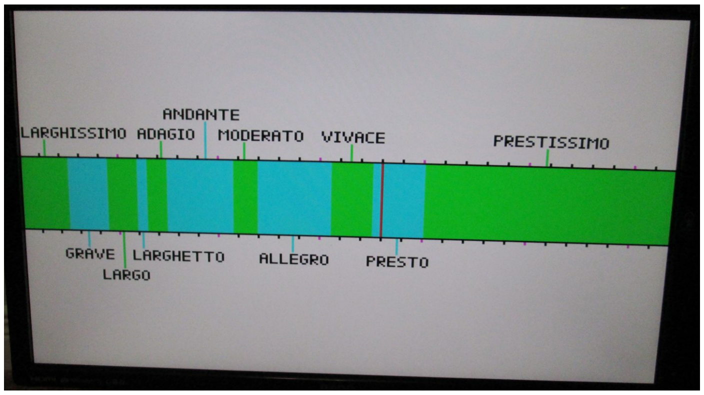
Report on Success

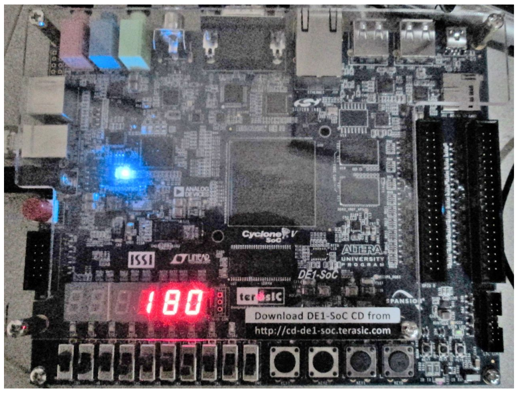
All three functionalities, including the interface with the hex displays and VGA output, work as intended.



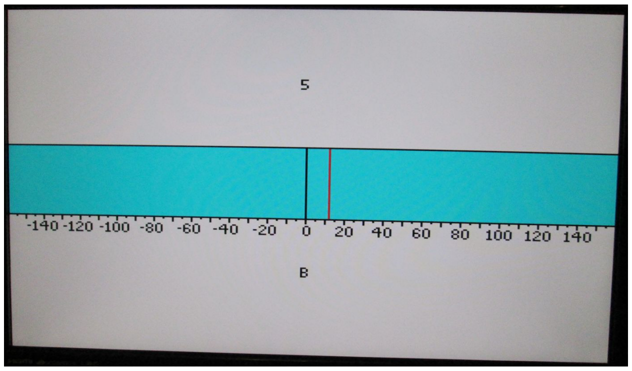


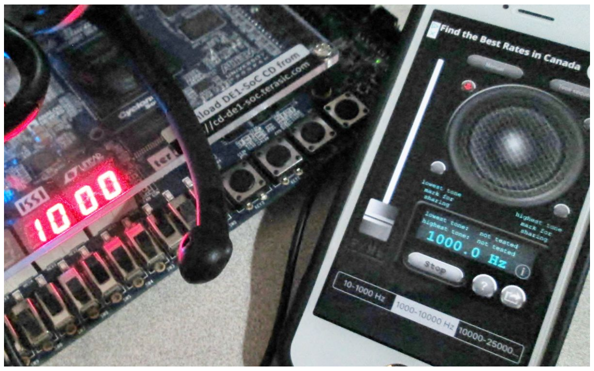
G sharp / A flat from octave 4 has been selected for the tone generator. The note G# is shown on the staff and also written out on the HEX displays.





The metronome is set to 180 bpm. That is in the "presto" range.





The phone is generating a 1000 Hz tone, which the tuner recognizes. The closest note is B5, which is 988 Hz, so the deviation is +12 Hz.

A major bug that was fixed was running out of memory. The FFT uses a lot of M10K memory. Initially, all background and sprite images were stored on M10K by default. The first attempted solution was to move all backgrounds to MLAB, but MLAB didn't enough room. The second solution was to make the note name and octave sprites much smaller, and only have one background in MLAB. This allowed the entire design to fit on the board.

Possible Improvements from Beginning

If the project was done again, we would use memory outside the FPGA itself, such as SDRAM, since memory was the biggest issue with the current design. We could also use a PS2 keyboard for input instead of keys and switches, since there could be many more input possibilities and more intuitive input methods (such as typing in a bpm value rather than inputting a binary number via switches). We could also generate sine waves instead of square waves to produce a cleaner tone with the tone generator.